Experimental Assessment of Routing for Grid and Cloud

Douglas O. Balen, Carlos B. Westphall, and Carla M. Westphall
Network and Management Laboratory
Federal University of Santa Catarina
Florianópolis – SC - Brazil
Emails: {douglasb, westphal, carlamw}@inf.ufsc.br

Abstract— **Grid and Cloud computing technologies are being applied as an affordable method to cluster computational power together. These structures aim to support service applications by grouping devices and shared resources in one large computational unit. However, the management complexity grows proportionally to the number of resources being integrated. The paper claims to address the problems of management, considering the routing problem in a particular context. An experimental assessment of routing for grid and cloud is presented. In addition, it introduces a proof-of-concept implementation and case study scenarios.**

Keywords - Grid and cloud computing; autonomic systems; routing; network management.

I. INTRODUCTION

Since the creation of the Internet, systems have become increasingly complex due to the scalability and availability requirements posed by several of today's Web services. The popularity of pervasive computing also contributes to increase this complexity as new portable devices are routinely released in the market and integrated into the Internet Cloud.

According to IBM [1], traditionally, networks and management systems are manually controlled processes which demand one or more human operators to manage all the computing systems aspects. In this environment, the operator is strongly integrated to the management process and his task is to execute low level system calls to solve imminent problems. Even though this kind of management, which keeps a human into the system, was appropriate in the past, it cannot cope with modern systems.

The need to connect many heterogeneous systems is one of the main necessities of grid and cloud computing, introducing new levels of complexity. Even though it is a complex environment, the configuration and management is done by humans. This characteristic makes this task slow and a subject of decision making problems. Even administrator errors can occur at this task. In order to avoid this problem a solution is needed in which the management does not need human intervention. Observing this scenario, a question emerges: How to manage efficiently and in an automated way a heterogeneous and complex environment, like grid or cloud?

In order to answer this question this work proposes an experimental assessment of routing for grid and cloud computing that supports autonomic computing paradigm. The system has self-management properties, and redefines the human operator's responsibilities, where their experience is used to define general objectives and polices to control the system instead of placing them in a decision making position.

The rest of this work is organized as follows: Section II provides some comments on autonomic computing, and Section III discusses grid and cloud computing. Section IV proposes an experimental assessment of routing for grid and cloud computing and Section V describes the implementation and tests performed.

II. AUTONOMIC COMPUTING

An autonomic system is able to regulate its own functional parameters without incurring changes in the main system objectives. This way an autonomic system can optimize the use of its resources even under stress conditions. As described by Horn [2], to achieve complete autonomy a system must implement four main characteristics: self-configuration, self-healing, self-optimization, and self-protection.

The autonomic elements (AE), considered to be like the bricks of a building, are the functional units of autonomic systems. They control the resources and offer services to the users and other AEs. They also manage the internal behavior and its relations with other elements of the system, like the policies established by humans or other AEs. The autonomic behavior of the whole system emerges from the numerous interactions between the autonomic elements. An autonomic element consists of one or more managed elements, linked to a single autonomic manager (AM) that controls the managed elements, as shown in Figure 1. The managed elements can be a hardware or software resource.

What differentiates an autonomic from a non-autonomic system is the presence of the autonomic manager. Between monitoring of managed elements and its external environment, the autonomic manager is able to build and execute plans based on the analysis of sent information, which removes the need for human intervention.

 ISBN:978-1-61208-113-7

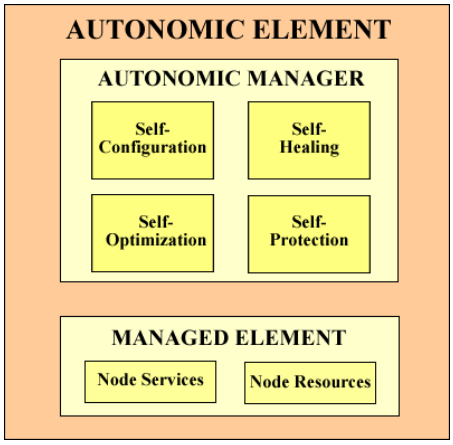

Figure 1. Autonomic Computing Element

III. GRID AND CLOUD COMPUTING

Grid and cloud computing solutions aim to simplify the access to resources (hardware and software) of a distributed system, some times giving the idea that they form a unique and powerful computer. This is achieved by techniques such as virtualization. Resource virtualization [3] minimizes the impact of heterogeneity by providing access to well defined interfaces or to work units in terms of virtual machines. Using this set of abstractions the user can connect several different devices on his network.

The middleware is the software layer between the operational system and the applications, which provides some services that are needed by the applications. It creates the grid environment and gives transparency to the applications. There are several projects in this field, including Globus [4], Gridbus [5], Legion [6], UNICORE [7], Alchemi [8], OurGrid [9] and Grid-M [10].

TABLE I
SOME CURRENT MIDDLEWARES

	Collaboration Support	Context Awareness Support	Resource Allocation Support	Dinamic Enviroument Support	Mobile Enviroument Support	Self-Management Support
Globus	■		■			
Legion	■		■			
Gridbus	■		■		■	
UNICORE	■		■			
Alchemi	■		■			
OurGrid	■		■			
Grid-M	■	■	■	■	■	

Looking at the features supported by these systems in Table 1, one can see that all listed middleware support collaboration and resource allocation. However, only two systems support execution on mobile environments. Only one provides context sensibility. None of them supports autonomic behavior. Due to a grid complexity, there is a need for middleware that supports autonomic.

There is no consensus about what exactly cloud computing is, but some characteristics are clearly repeated. It is a new distributed computing and business paradigm. It can provide computing power, software and storage resources, and even a distributed data center infrastructure on demand. To make these characteristics viable, it uses existing technologies, such as virtualization, distributed computing, grid computing, utility computing and the network infrastructure provided by the Internet. In this work, we are considering cloud computing, using our middleware Grid-M [10], developed by the Laboratory on Networks and Management.

IV. ROUTING FOR GRID AND CLOUD COMPUTING

Pervasive computing is a paradigm that aims to provide a computing environment anywhere through the use of virtualization of information, services and applications.

A middleware capable of supporting this new computational environment must offer large scale distributed computing that permits to integrate sensors and mobile devices, always taking into consideration the dynamics of the environment and the context sensibility.

The only middleware from those examined that presents these characteristics is the Grid-M [10]. However, similarly to others, it does not offer autonomic behavior. The computational grids are known as a dynamic and heterogeneous computational environment, even though, the configuration of these environments is done manually and susceptible to slow decision making or errors of the administrators. In order to avoid this problem a solution is needed to take the responsibility away from the human administrators.

This work proposes a system for this kind of environment, offering the opportunity to create a grid and cloud computing with autonomic management.

A. *Related Work*

The system proposed intersects with fields that are being the target of continuous academic research such as autonomic systems, and grid and cloud computing. However, the union of these initiatives is still new and related work with the same focus is scarce. Some of the projects in this area are:

- Liu et al. [11] proposes an autonomic architecture to manage the heterogeneity and dynamics of the grid environments. This architecture allows the behavior of services and applications and its interactions to be specified and adapted according to the high-level rules. Everything is based on the requisites, states and execution context of the applications;
- Beckstein et al. [12] presents the SOGOS architecture aimed to support self-organization in computational grids. This is allowed to work with dynamic environments through semantic information

 ISBN:978-1-61208-113-7

(metadata) that describes the involved organizations, roles, rights of the participating agents and how they interact to solve the problem. The decisions are based on the metadata;

- Brennand et al. [13] presents the AutoMan, a system which has the objective of offering certain levels of automatic management to the computational grids in pairs. Beyond this scope, it tries to optimize the usage of resources on the grid, simplifying the management activities at the same time;
- Buyya et al. [14] defines Cloud computing and provides the architecture for creating market-oriented Clouds by leveraging technologies such as Virtual Machines (VMs);
- Xiao et al. [15] adapts web pages to small screen devices. In addition, as the limited computing ability and capacity of storage of wireless handheld devices, it is extremely challenging to deploy existing web page adaptation engine. By utilizing the large computing and storage resource capabilities of cloud computing infrastructures, a new wireless web access mode is proposed; and
- Vieira et al. [16] shows a solution for intrusion detection in grid and cloud computing environment in which audit data is collected from the cloud and two intrusion detection techniques are applied.

B. *Autonomic Manager*

What allows a system to be called autonomic is a presence of an autonomic manager. Through the monitoring of managed elements and their external environment, the autonomic manager is able to build and execute plans for implementation, based on the analysis of sent information. Therefore, the autonomic manager is responsible for ensuring self-management, achieved when all its sub-areas (self-configuration, self-regeneration, self-optimization and self-protection) are guaranteed.

For this purpose, this paper suggests that the manager is composed of some components, responsible for monitoring the data sent by the managed elements and others elements of the autonomic grid, analyze them, plan actions according to their objectives and implement these actions, thus achieving a high degree of autonomy.

C. *Routing among the nodes*

The number of mobile devices is constantly changing, which can result in big changes in the overall system. For the interconnection among the devices, it is essential to keep the routing table consistent. The Routing Table Management component has the goal of detecting routing inconsistencies, but it cannot directly manipulate the routing table. The latter is done by the grid's routing algorithm.

The system proposed here implements two routing algorithms: one is based on the direct interconnection with a neighbor node, and the other is based on the interconnection among all nodes.

In grids, every element has its own routing table that contains the destination (node name) and a metric (the distance until the next element in hops). On the first algorithm, each node connects to the neighbor node only. Thus, the route to the neighbor node becomes a default route (gateway) to the other elements in the grid. For example, when an element wants to request a service, it sends a request to the gateway, and the gateway is responsible for forwarding the request to the others nodes connected to it. This process is repeated until the destination receives the request.

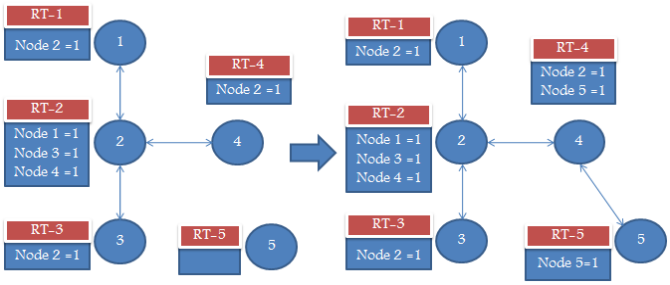

Figure 2. Routing algorithm based on the direct interconnection to the neighbor node.

Figure 2 illustrates how that algorithm works. On the left side, node 5 is out of the grid, thus the other elements cannot connect to it. As soon as node 5 joins the grid through node 4, the latter includes a route to node 5 with metric 1, i.e., directly connected.

The other algorithm is a little different. As an element joins the grid, all the other elements add a direct route to it (metric 1). This makes the whole grid to be seen as a complete graph. The propagation of the information about a node joining or leaving the grid is coordinated by this same algorithm in an autonomic way. When all the nodes discover the topology changes, we have reached the convergence.

Figure 3 illustrates this situation. At first, node 5 is out of the grid. Note that all other elements are directly connected (metric 1). Then, node 5 is included. It does not matter knowing which node it is connected to, because the distance among all elements is the same. The first node to notice its join-request is going to add a direct route to it, sends its actual routing table, and finally informs all the other elements that there is a new node in the grid.

 ISBN:978-1-61208-113-7

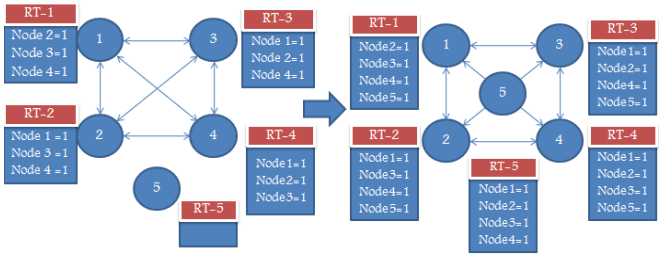

Figure 3. Routing algorithm based on the complete connection among nodes

It is the responsibility of the grid's manager to decide which algorithm to use. Remark that it is not possible to use both algorithms at once, all nodes must use the same algorithm.

V. IMPLEMENTATION AND TESTS

To this point, this paper described the theory upon which the proposed system was based, the architecture details, its components and interactions, and the routing algorithms. To test it, we have implemented it on Grid-M [10]. Among the main benefits of the Grid-M middleware are: it is open source, it is easy to deal with small devices, it has a friendly API and it is portable [10].

This section shows the results of a few quantity tests performed during the implementation with the purpose of showing the proposed system efficiency in different use situations.

A grid of 30 nodes was created. These devices are personal computers with an Intel Core Duo 1.66Ghz CPU, 2GB of RAM memory and running Window XP. All devices ran the same programs.

A. *Convergence Time*

Here, we do three separated tests for the two kinds of algorithms to test the convergence time. To a routing protocol, convergence time means the time it takes for all the routing tables to be updated when there is a change on the topology (e.g., when a node joins the grid).

At the beginning, we thought the convergence time would be a bottleneck, especially on the algorithm which all nodes are directly connected since all routing tables are spread among all nodes.

Analyzing Figure 4 though, which shows the convergence time of the algorithm based on the direct interconnection to the neighbor node, we notice that the convergence time is really small and almost constant (varying between 10ms and 14ms). This happens because the only processing needed is the inclusion of the neighbor's route in the routing table. No data about a joining node is passed along. The time was taken when a new element joined the grid. The elements were added in the following manner: node 2 connects to node 1, node 3 connects to node 2, node 4 connects to node 3, and successively.

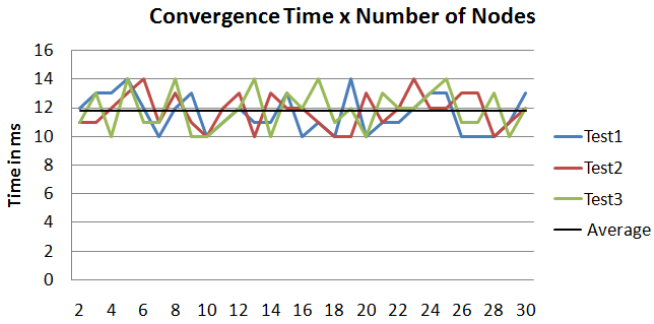

Figure 4. Convergence time – Algorithm based on the direct interconnection to the neighbor node

On the other hand, on the other algorithm, when a node joins the grid, all elements' routing tables are updated with the new information. The convergence time of this algorithm is shown on Figure 5. The data was obtained the same way as the previous test.

Figure 5 shows that the lowest convergence time was achieved on test 2, after the insertion of node 6, and the highest convergence time was achieved on test 2 as well, after the insertion of node 10. As you can see, as more nodes get in the grid, the convergence time increases, but on an ease pace (the average time at the beginning was 138ms and at the end it was 144ms). As the convergence time was still low in this case, we chose this algorithm because its response time during tasks executions is a lot lower.

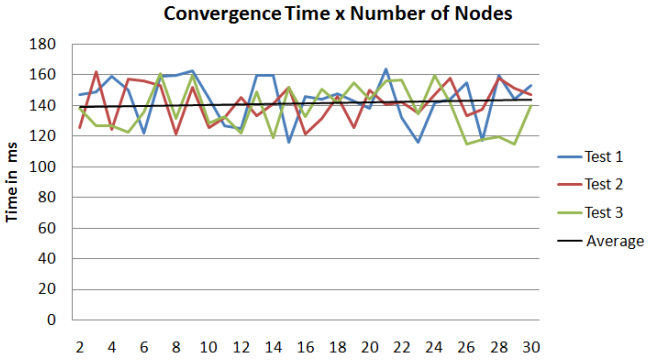

Figure 5. Convergence time – Algorithm based on the complete connection among the nodes

 ISBN:978-1-61208-113-7

B. *Response time of the tasks execution*

Another important point is the response time of a service request. The response time refers to the sum of two distinct tasks: the service search time and the task execution time.

The task search time is the time a requester spent to search for a determined service in the grid to find who has it and who has the best available resource percentage. So, at the end of the search, we have the best candidate for the execution and a task is created with him being the destination. All the process of search and request redirections is managed and controlled by the Autonomic Manager.

With the intention of testing the response time of the service execution requests, we have used the same structure of the previous test (30 equal nodes). The test consists in the node 1 request a service to the grid. The only node that has it is the node 30.

We would like to clarify that the response time depends on the routing algorithm type utilized. As we use the algorithm based on the direct interconnection with the neighbor node, the search takes longer if we compare it to the algorithm based on the complete interconnection among nodes. This happens because the latter has a complete view of the topology. Therefore, the search in all nodes can be done in parallel (by using threads). Case we use the first algorithm, the search request must pass through the intermediate nodes before getting to its destination. The test results using both algorithms are shown on Figure 6.

As expected, the response time of the algorithm based on the restrict connection to the neighbor node is longer than the other one.

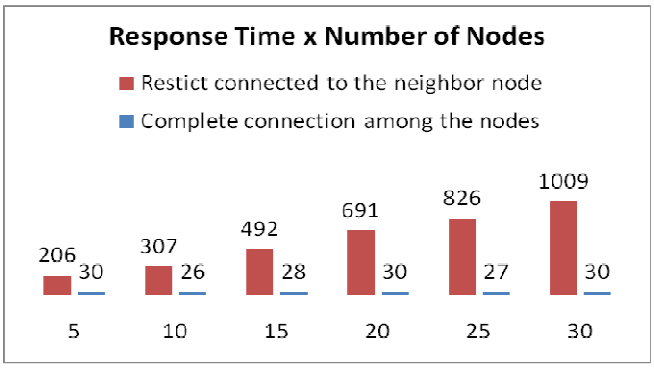

Figure 6. Response time results

C. *Efficiency of the services replication*

On the previous test, the node 1 requested a service to the grid. When the search was done, it was verified that only node 30 offered the determined service. As it was a test and we knew that there was only one requester, we discard the possibility of the node 30 being a bottleneck due to it being overloaded. However, what would happen if the other 29 nodes requested the same service? On this case, there would be the possibility of the node 30 not being able to answer to all requests on the best possible way, lowering the performance of the grid. At this time the node 30, aware that he is overloaded, would send a replication request to find an available element that offers the same service. Note that the replication is necessary only once. After that, the node that received the service will start answering to other requests about the same service.

Figure 7 shows the resources used by 4 elements during the tests. We got this information from the Grid-M logs.

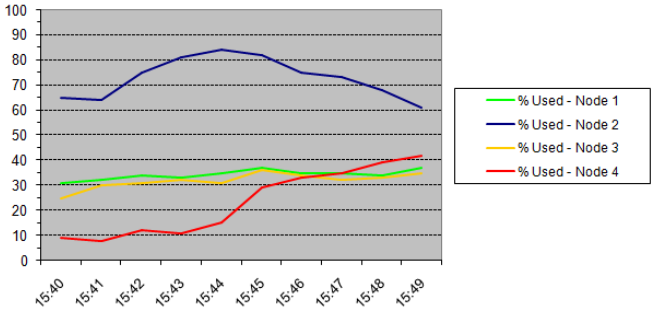

Figure 7. The resources utilized by the nodes and the services replications

On this test, nodes 1 and 3 make requests all the time to a service that initially only node 2 provides. After a while, node 2 becomes overloaded (the free resources percentage gets lower than 18%) and then a service replication occurs, from node 2 that has the service, to node 4 that was the node which had more free resources at that time. After that, the service requests are answered by node 4 as well, distributing the processing of these requests. Analyzing the chart (Figure 7), we observe that the algorithm eliminated the eminent saturation of the node 2 and the possible creation of a bottleneck in the grid.

VI. CONCLUSION

In this paper, we have proposed an experimental assessment of routing for grid and cloud computing. The convergence time of the algorithm based on the direct interconnection to the neighbor node is really small and almost constant. As expected, the response time of the algorithm based on the restrict connection to the neighbor node is longer than the other one. The big question to be answered was: How to make a heterogeneous environment and with huge complexity, like grid and cloud computing, not being managed manually, which is inefficient? The solution proposal is the creation of autonomic elements acting as intelligent agents, capable of feel the environment where they are and act the same according to pre-defined policies.

 ISBN:978-1-61208-113-7

REFERENCES

[1] IBM-Corporation. An architectural blueprint for autonomic computing. *http://www.ibm.com/developerworks/autonomic/library/ac-summary/ac-blue.html*

[2] P. Horn. Autonomic computing: IBM's perspective on the state of information technology. Technical report, *International Business Machines Corporation*, Armonk, NY, USA, 2001.

[3] J. Joseph and M. Ernest. Evolution of grid computing architecture and grid adoption models. *IBM Systems Journal*, 2004, 43(4).

[4] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Internacional Journal of Supercomputer Applications*, 1997, 11(2), pp. 115–128.

[5] R. Buyya. Market-oriented grid computing and the gridbus middleware. *16th International Conference on Advanced Computing and Communications*, 2008. ADCOM 2008.

[6] A. Grimshaw and A. Natrajan. Legion: Lessons learned building a grid operating system. *Proceedings of the IEEE*, 2005, 93(3), pp. 589–603.

[7] UNICORE. UNIform Interface to Computer Resources. http://www.unicore.eu/ (last access on Dec. 2010).

[8] A. Luther, R. Buyya, R. Ranjan, and S. Venugopal. Alchemi: A .net-based grid computing framework and its integration into global grids, pp. 1-17, 2005. (informal publication). http://www.cloudbus.org/papers/Alchemi.pdf

[9] F. Brasileiro, E. C. de Araujo, W. Voorsluys, M. Oliveira, and F. Figueiredo, "Bridging the High Performance Computing Gap: the OurGrid Experience," ccgrid, pp.817-822, Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07), 2007.

[10] H. A. Franke, C. Rolim, C. B. Westphall, F. Koch, and D. O. Balen. Grid-M: Middleware to integrate mobile devices, sensors and grid computing. *The Third International Conference on Wireless and Mobile Comunications – ICWMC* 2007.

[11] H. Liu,V. Bhat, M. Parashar, and S. Klasky. An autonomic service architecture for self-managing grid applications. InGRID'05: *Proceedings of the 6th IEEE/ACM Internation Workshop on Grid Computing*, 2005.

[12] C. Beckstein, P. Dittrich, C. Erfurth, D. Fey, B. Konig-Ries, M. Mundhenk, and H. Sack. Sogos-distributed meta level architecture for the self-organizing grid of services. *In MDM'06: Proceedings of the 7th International Conference on Mobile Data Management*, 2006.

[13] C. Brennand, M. Spohn, A. Souza, G. Ferreira, D. Candeia, G. Germoglio, and F. Santos. Automan: Autonomic Management on Ourgrid. *V Workshop for Grid Computing and Aplications,* 2007.

[14] R. Buyya, C. S. Yeo, and S. Venugopal, Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. *10th IEEE International Conference In High Performance Computing and Communications*, 2008.

[15] Y. Xiao, Y. Tao, and Q. Li, A New Wireless Web Access Mode Based on Cloud Computing. *Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 2008.

[16] K. Vieira, A. Schulter, C. B. Westphall, and C. M. Westphall, "Intrusion Detection for Grid and Cloud Computing". IEEE IT Professional Magazine. V.12 (4). pp. 38-43. 2010.

 ISBN:978-1-61208-113-7